Hierarchies

CS 4620 Lecture 10

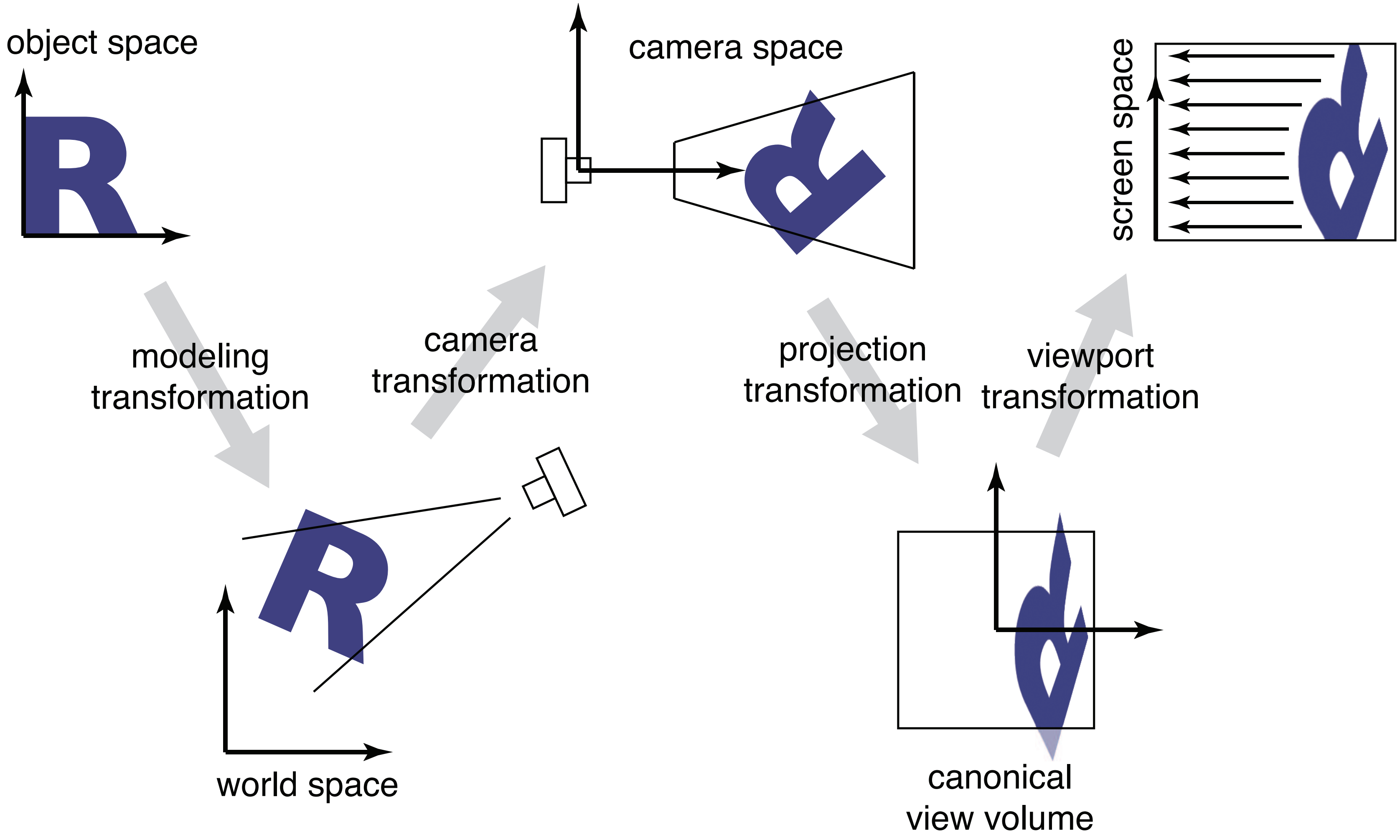
Announcements

- Released a GPU diagnostic

- A2 due this week
  - Demos on Monday (like last time)
  - Demo sign ups will be up shortly

Pipeline of transformations

- Standard sequence of transforms



© 2015 Kavita Bala
w/ prior instructor Steve Marschner •

Coordinate frame summary

- Frame = point plus basis
- Frame matrix (frame-to-canonical) is

$$F = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$

- Move points to and from frame by multiplying with *F*

$$p_e = Fp_F \quad p_F = F^{-1}p_e$$

- Move transformations using similarity transforms

$$T_e = FT_FF^{-1} \quad T_F = F^{-1}T_eF$$

Rigid motions

- A transform made up of only translation and rotation is a *rigid motion* or a *rigid body transformation*
- The linear part is an orthonormal matrix

$$R = \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$$

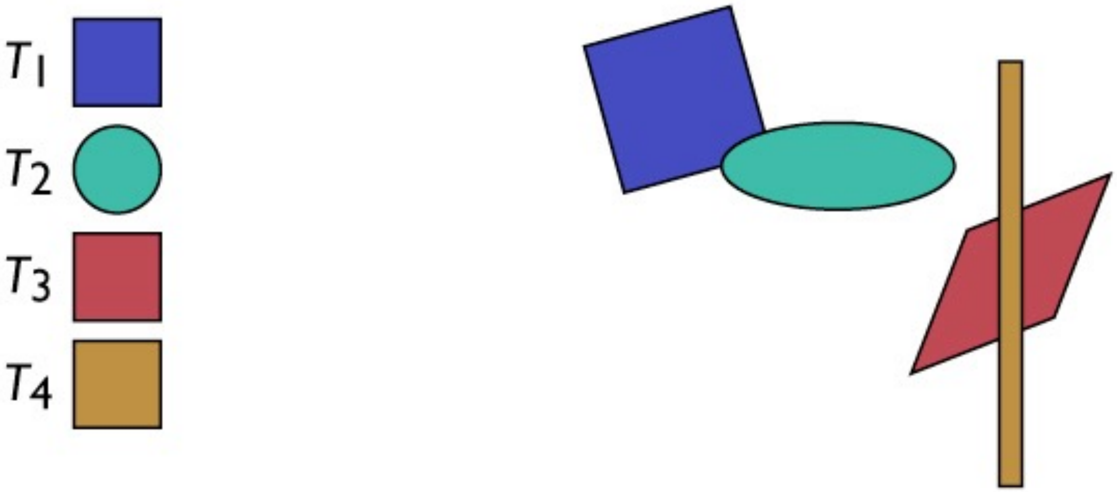
- Inverse of orthonormal matrix is transpose
  - so inverse of rigid motion is easy:

$$R^{-1}R = \begin{bmatrix} Q^T & -Q^T\mathbf{u} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$$

Hierarchies and Transformations

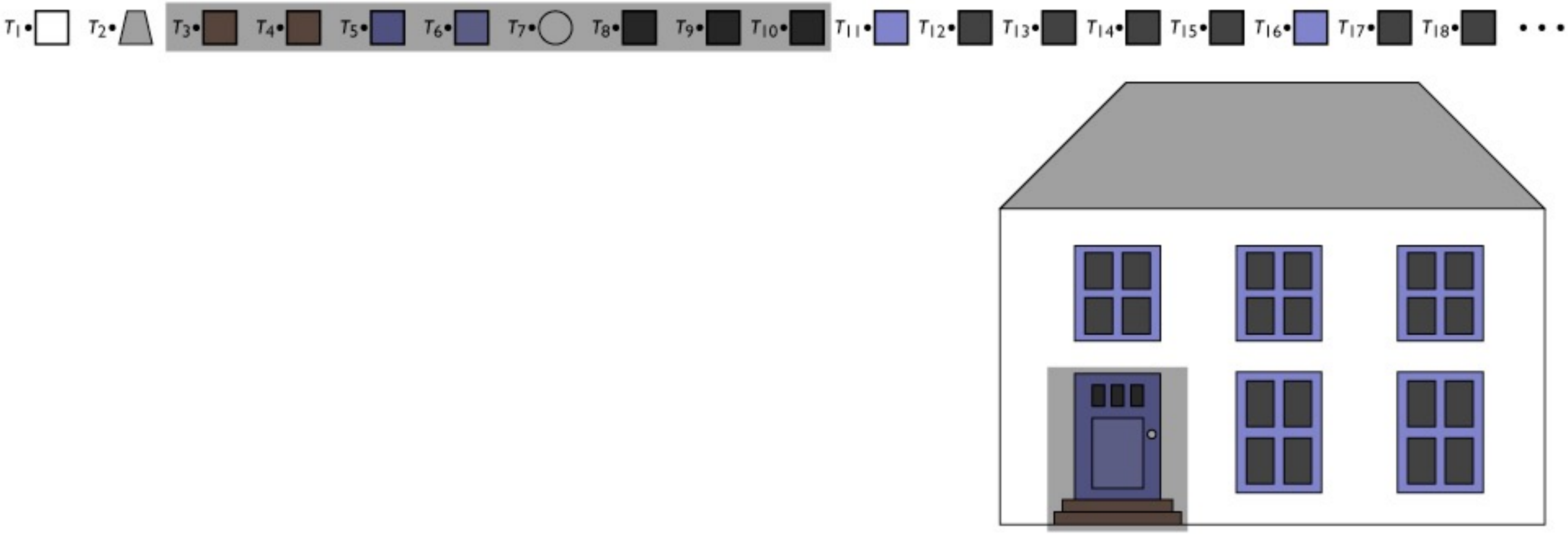
Data structures with transforms

- Representing a drawing (“scene”)
- List of objects
- Transform for each object
  - can use minimal primitives: ellipse is transformed circle
  - transform applies to points of object



Example

- Can represent drawing with flat list
  - – but editing operations require updating many transforms

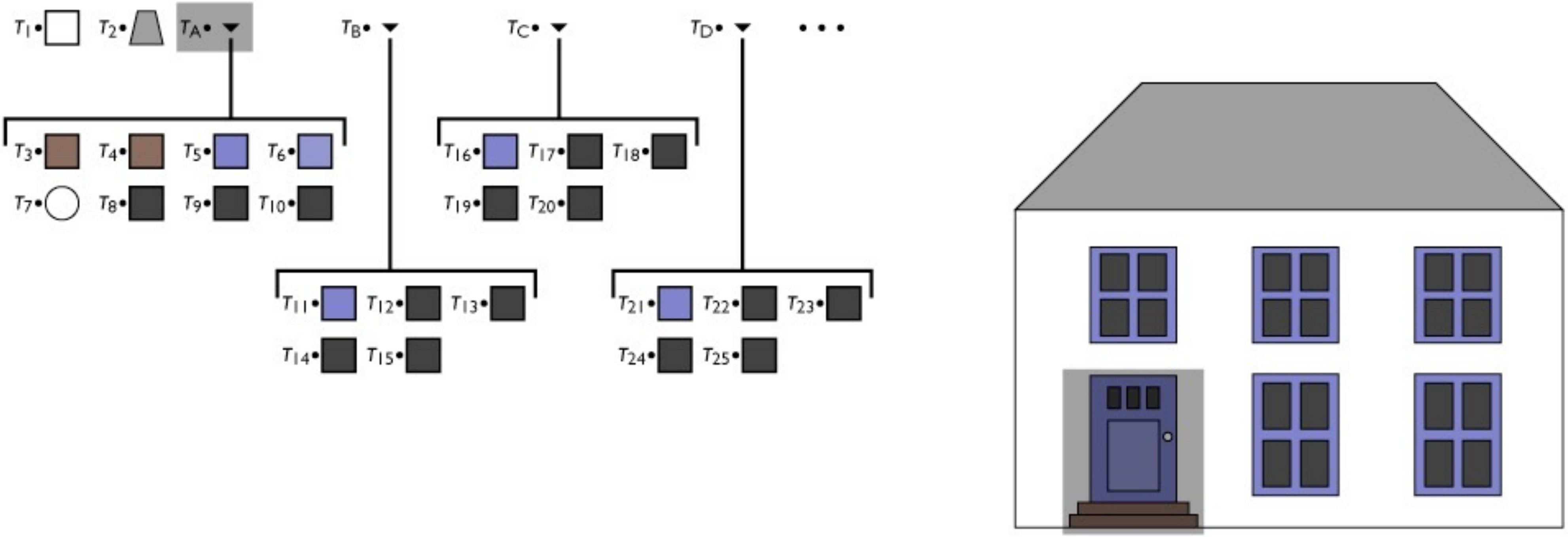


Groups of objects

- Treat a set of objects as one
- Introduce new object type: group
  - contains list of references to member objects

Example

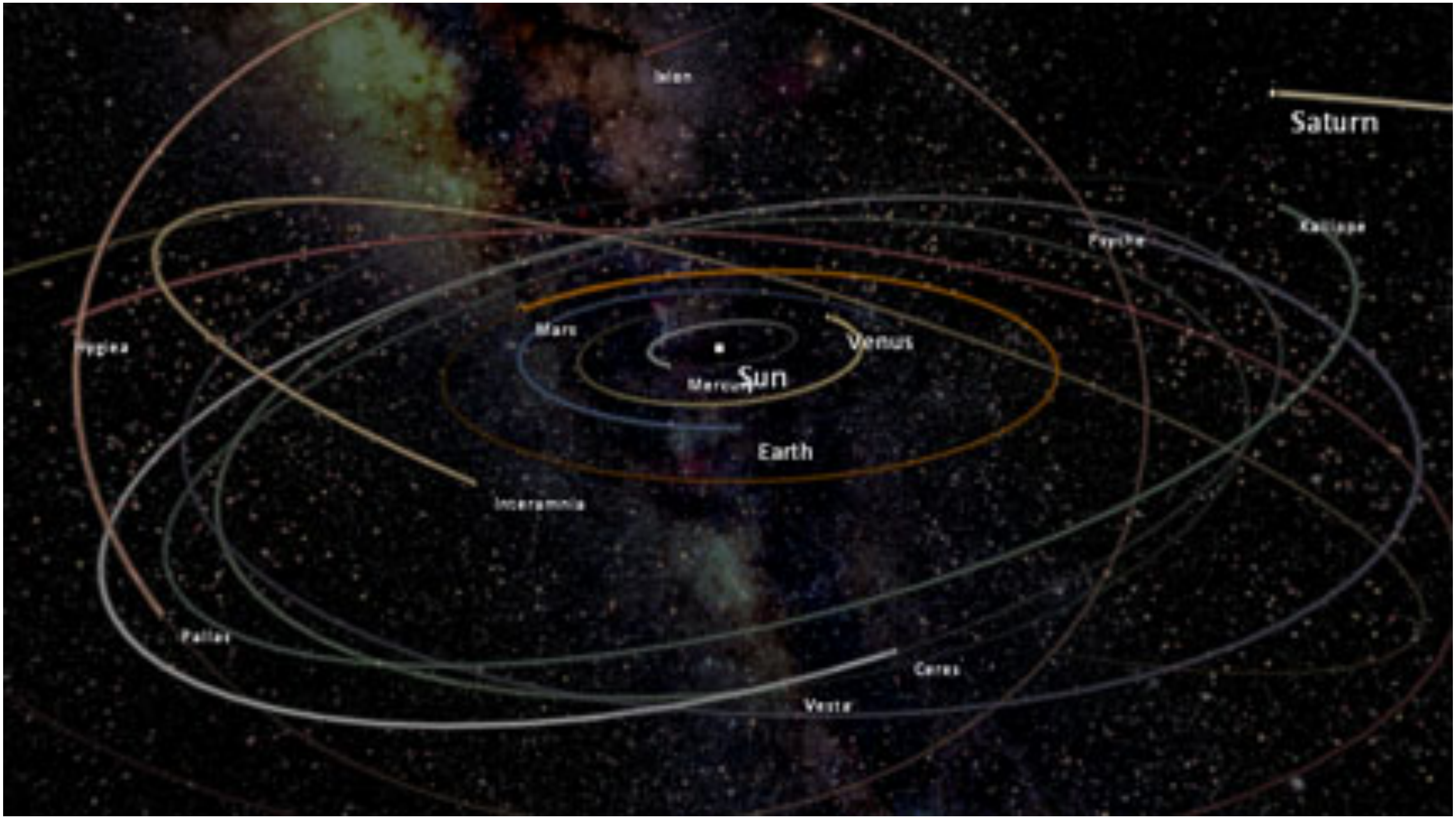
- Add group as a new object type
  - lets the data structure reflect the drawing structure
  - enables high-level editing by changing just one node



Groups of groups: hierarchies

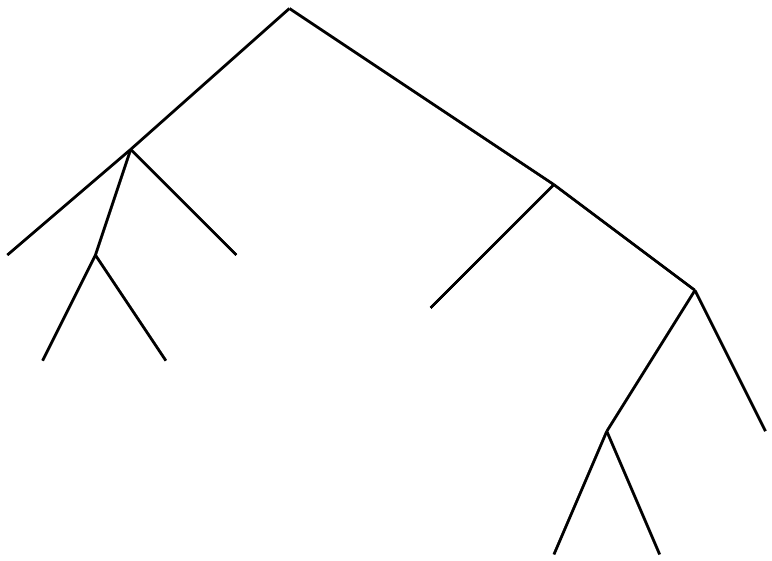
- This makes the model into a tree
  - interior nodes = groups
  - leaf nodes = objects
  - edges = membership of object in group

- Hierarchies
  - Important for modeling and animation
  - Models have parts. Parts have convenient coordinate system
  - E.g., moon around earth, earth (+moon) around sun, sun around galaxy center, galaxies spinning out in the universe


Ixion
Saturn
Kalliope
Psyche
Hygiea
Mars
Venus
Sun
Mercury
Earth
Interamnia
Pallas
Ceres
Vesta

The Scene Graph (tree)

- Grouping applied hierarchically
- Scene graph: name for various kinds of graph structures (nodes connected together) used to represent scenes
- Simplest form: tree
  - every node has one parent
  - leaf nodes are identified with objects in the scene



© 2015 Kavita Bala
w/ prior instructor Steve Marschner • 13

Concatenation and hierarchy

- Transforms associated with nodes or edges
- Each transform applies to all geometry below it
  - want group transform to transform each member
  - members already transformed—concatenate

Concatenation and hierarchy

- Transforms associated with nodes or edges
- Each transform applies to all geometry below it
  - want group transform to transform each member
  - members already transformed—concatenate
- Frame transform for object is product of all matrices along path from root
  - each object's transform describes relationship between its local coordinates and its group's coordinates
  - frame-to-canonical transform is the result of repeatedly changing coordinates from group to containing group

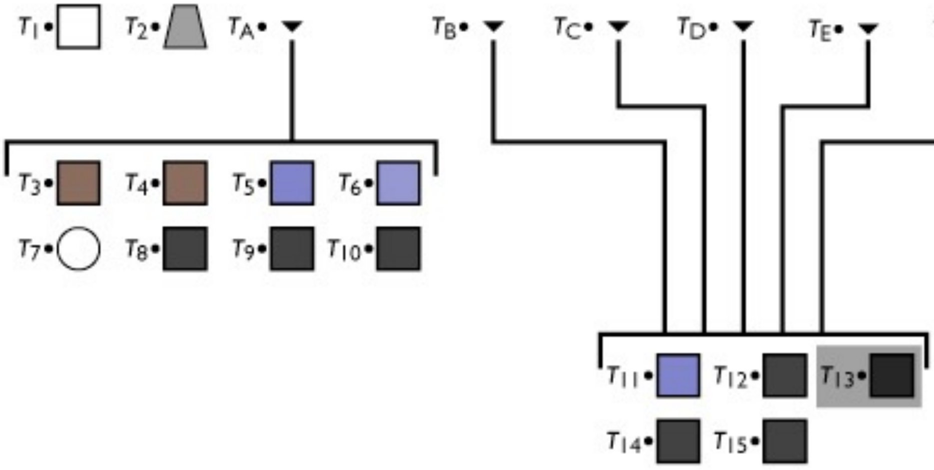
Large scenes

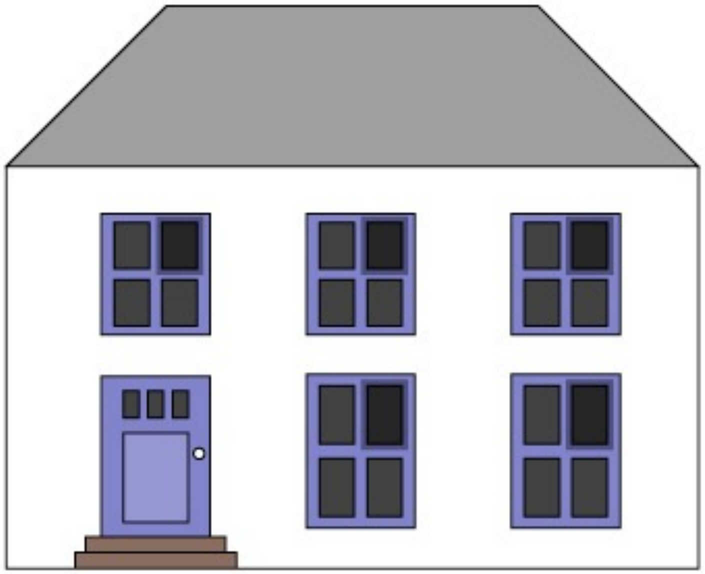
- Lot of replicated units

- Instancing
  - Simple idea: allow an object to be a member of more than one group at once
  - transform different in each case
  - leads to linked copies
  - single editing operation changes all instances

Example

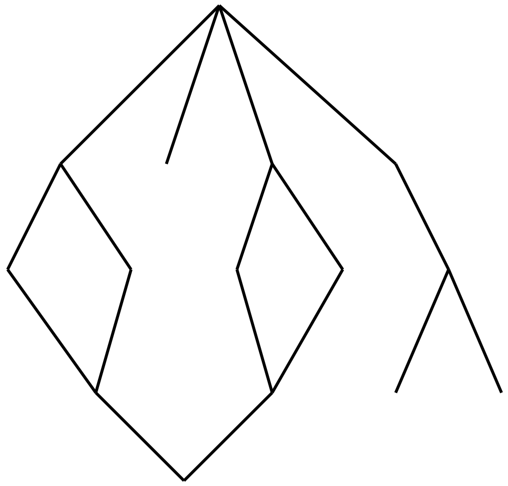
- Allow multiple references to nodes
  - reflects more of drawing structure
  - allows editing of repeated parts in one operation





The Scene Graph (with instances)

- With instances, there is no more tree
  - an object that is instanced multiple times has more than one parent
- Transform tree becomes DAG
  - **d**irected **a**cyclic **g**raph
  - group is not allowed to contain itself, even indirectly
- Transforms still accumulate along path from root
  - now *paths* from root to leaves are identified with scene objects



© 2015 Kavita Bala
w/ prior instructor Steve Marschner

Implementing a hierarchy

- Object-oriented language is convenient
  - define shapes and groups as derived from single class

```
abstract class Shape {
   void draw();
}

class Square extends Shape {
   void draw() {
      // draw unit square
   }
}

class Circle extends Shape {
   void draw() {
      // draw unit circle
   }
}
```

Implementing traversal

- Pass a transform down the hierarchy
  - before drawing, concatenate

```
abstract class Shape {
   void draw(Transform t_c);
}

class Square extends Shape {
   void draw(Transform t_c) {
      // draw t_c * unit square
   }
}

class Circle extends Shape {
   void draw(Transform t_c) {
      // draw t_c * unit circle
   }
}
```

```
class Group extends Shape {
   Transform t;
   ShapeList members;
   void draw(Transform t_c) {
      for (m in members) {
         m.draw(t_c * t);
      }
   }
}
```

© 2015 Kavita Bala

Basic Scene Graph operations

- Editing a transformation
  - good to present usable UI
- Getting transform of object in canonical (world) frame
  - traverse path from root to leaf
- Grouping and ungrouping
  - can do these operations without moving anything
  - group: insert identity node
  - ungroup: remove node, push transform to children

Adding more than geometry

- Objects have properties besides shape
  - color, shading parameters
  - approximation parameters (e.g. precision of subdividing curved surfaces into triangles)
  - behavior in response to user input
  - …
- Setting properties for entire groups is useful
  - paint entire window green
- Many systems include some kind of property nodes
  - in traversal they are read as, e.g., "set current color"

Scene Graph variations

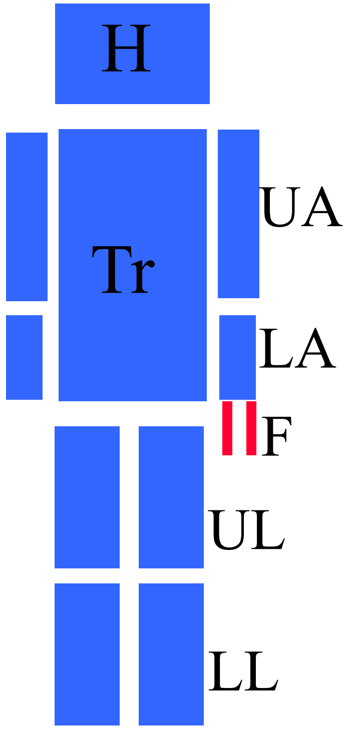
- Where transforms go
  - in every node
  - on edges
  - in group nodes only
  - in special Transform nodes
- Tree vs. DAG
- Nodes for cameras and lights?

Hierarchy Example

- Articulated body
- Every object has local frame of reference

- T (UA to Tr) T(LA to UA) T (F to LA)

- Think of applying it to a point
- Think of applying it to the coordinate system



© 2015 Kavita Bala
w/ prior instructor Steve Marschner

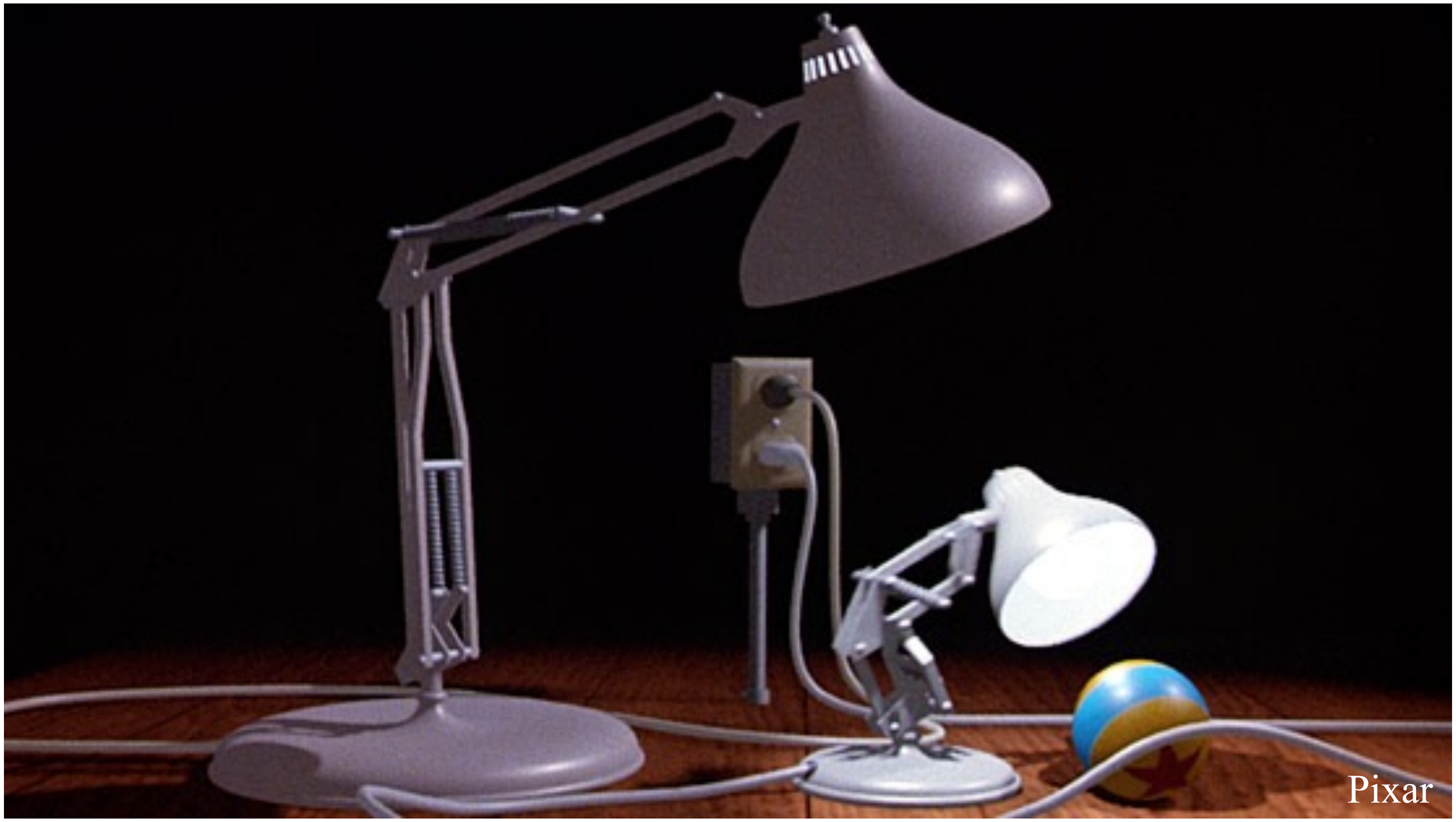
In OpenGL

- Have a stack of transforms

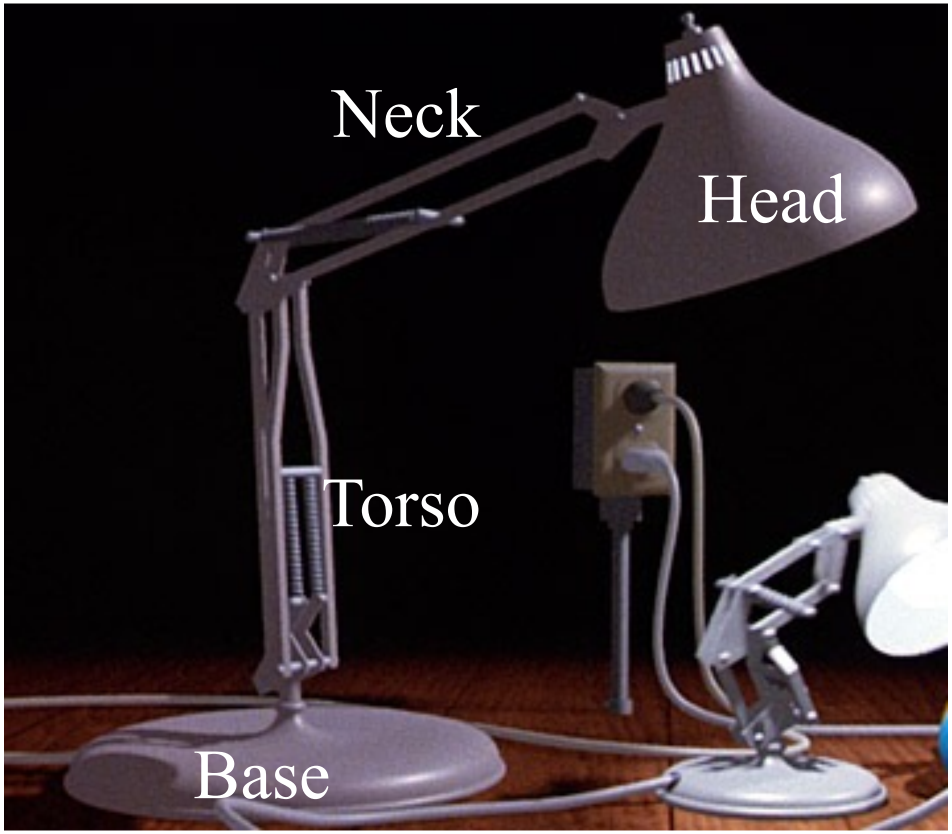
- You push and pop transforms on the stack
- glPushMatrix, glMultMatrix, glPopMatrix

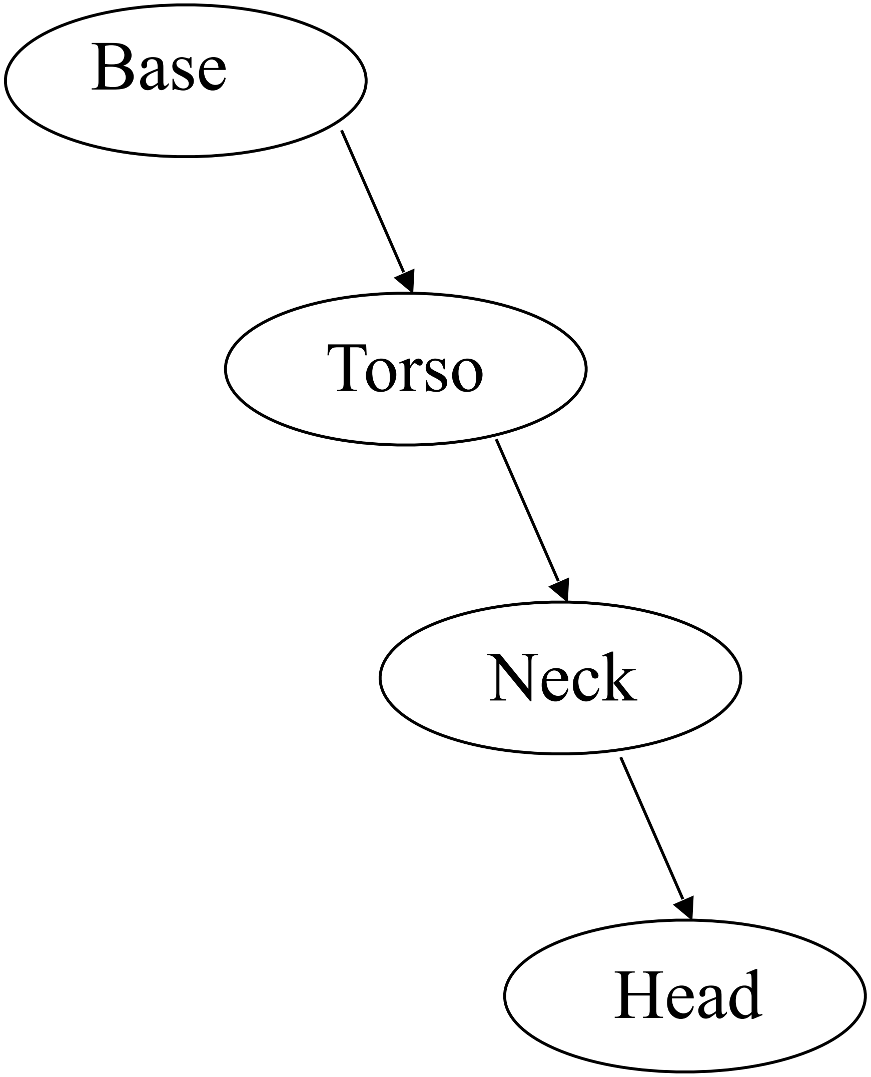
- Depth first traversal
- Start with identity
- Push as you go down, pop as you go up

Pixar's Lamp



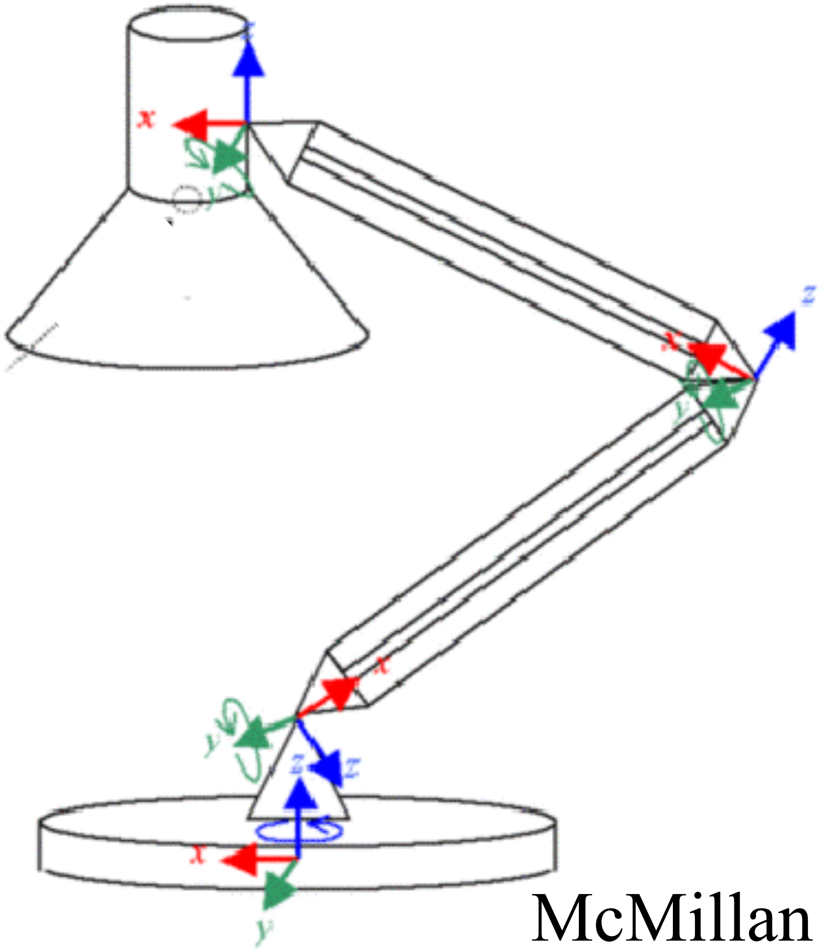
Hierarchy





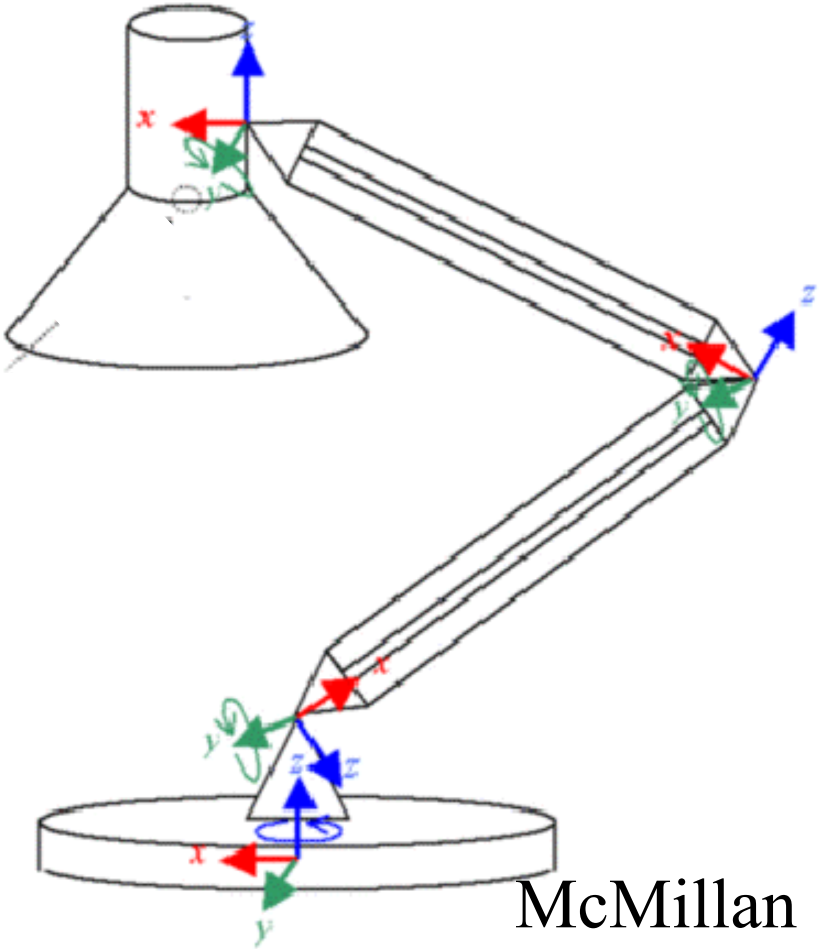
© 2015 Kavita Bala
w/ prior instructor Steve Marschner

Local Coordinate Systems



McMillan

Transforms for Head



McMillan

- Translate (0, 0, 2.5)
- Rotate (-120, 0, 1, 0)
- Translate (12, 0, 0)
- Rotate (65, 0, 1, 0)
- Translate (12, 0, 0)
- Rotate (30, 0, 1, 0)